Fermilab

TM-1343
2330.000

A BEGINNERS GUIDE TO CBASIC

Randy Zamin
(with the help of Bill Higgins and Rick Johnson)

August 1985

A Beginner's Guide To CBASIC

Written by: Randy Zamin
(with the help of Bill Higgins and Rick Johnson)

CBASIC Cookbook Table of Contents

I.	Logging On	2
II.	Looking at Directory	2
III.	Listing a Program	3
IV.	Running a Program	4
V.	Saving a Program	5
VI.	Deleting a Program	5
VII.	Copying a Program From Another Acct.	5
VIII.	CBASIC Commands	6
IX.	Selected Statements and Examples	7

Programming in CBASIC

There is not a CBASIC programming manual, and much of what is known about CBASIC is strictly folklore (Higgins, 1985). This is a "collection" of the basic knowledge for getting started in this language.

I. Logging On**

```
[1] LOG accountname/password (cr)
```

Once you are logged on, you may begin programming (or running programs) in your own account. If you are not logged onto your account, you are in the default account called "Viewer". You may still program in this account.

II. Looking at the Directory*********************************

It may be wise to first explain the format of the files and UIC's. The actual file name generally consists of the following format:

```
[group,user]filename.EXT;version
```

The brackets contain the user's identification code (UIC). The default for this is the UIC that is presently logged on. The filename is the name of the file that you wish to retrieve. The ".EXT" is the extension of the file. In other words, it describes the type of file. Their are 4 ~~t~~many types of extensions, but those used most frequently are ".BAS" (CBASIC), ".PAG" (Page), and ".DAT" (Data). The version is the saved modification of the stated file. The default for the version is the most recent on file.

```
[1] DIR (cr)
```

This will list all files in the UIC on which you are logged. DIR [UIC] will list all files under the specified UIC (Users Identification Code).

If you are looking for a particular file, you must add the following to the command:

```
[1] DIR programname.EXT
```

Ex.

```
[1] DIR SHIFT.BAS;* (cr)
```

This will give you a directory of all versions of the CBASIC program SHIFT.

The character * can be used as a wildcard in any of the stated parameters. The following example illustrates this idea again.

Ex.

```
[1] DIR [*,*]SHIFT.*
```

This will look for any file named SHIFT in all accounts.

III. Listing a Program*********************************

When you are looking for a CBASIC program, you must use the extension ".BAS". When loading the program, this rule does not apply.

Press (cr) until you get the prompt

1>
--

```
        [1] OLD filename (cr)
        [2] LIST
```

If you wish to stop the scroll:

```
        [3] Press CTRL-S
```

When you want to continue:

```
        [4] Press CTRL-Q
```

To list again, repeat steps starting at [2].

Ex. The following steps will call a program, list it all the way through, and then just list lines 1-20.

```
        [1] OLD RAMP (cr)
        [2] LIST (cr)
        [3] LIST 1-20 (cr)
```

IV. Running a Program:**********************************

Press (cr) until you get the prompt

1>
--

```
        [1] OLD filename (cr)
        [2] RUN (cr)
```

or

```
        [1] RUN filename (cr)
```

You may also RUN a program from another account (as long as you know the UIC of that account). To find the UIC of the account that you are presently in, simply type "SHOW UIC". If you would like to see who has been using your account, type "SHOW USERS".

Ex.

```
        [1] RUN [UIC of other account]filename (cr)
```

V. Saving a Program:***************************************

In order to save the program into the account that you are presently on, you must do the following:

```
        [1] SAVE filename (cr)
```

VI. Deleting a Program:**********************************

```
        [1] PURGE filename.EXT (cr)
```

This will delete all versions of your file except the latest

The following step will delete all versions of a filename:

```
        [1] UNSAVE filename (cr)
```

VII. Copying a Program From Another Account**************

Copying a program from some other account requires 2 things:

1) You must be in the account at which you are copying the program.

2) You must know the UIC from the account that you are taking the program.

```
        [1] OLD [UIC of other account]filename (cr)
        [2] SAVE filename (cr)
```

VIII. CBASIC Commanus*********************************

The following commanus are those found to be most useful to a beginning programmer.

BYE	Closes open files, exits CBASIC and returns the console to the default account.
CLEAR	Deletes all variables.
DEL	Deletes all lines from program in resident memory.
DEL m	Deletes line m from a program.
DEL m-n	deletes lines m through n.
DEL l, m-n	Deletes line l and m through n.
DIR	Lists the entire directory of the user.
DIR [UIC]	Lists the entire directory of the specified UIC.
LIST	Displays program statements sequentially.
LIST m	Displays line m of the program.
LIST m-n	Displays program lines m through n.
LIST l-m, n	Lists the desired program lines l through m and n.
OLD filename	Loads the specified program into memory.
RUN	Commences execution of program in memory.
RUN filename	Loads the program into memory and exe executes it.
UNSAVE filename.EXT;version	Deletes some version of the file on disk.

IX. Selected Statements and Examples

Before we move into the various statements and how they are used, it may be wise to first explain how a computer executes a program. Simply stated, the computer takes commands in sequential order (unless otherwise stated in the program), and executes one line at a time. In order for the computer to know which line to execute next, the program must be systematically ordered. In CBASIC, this is done by numbering each line of the program. The following example demonstrates this ordering.

Ex. 1

```
20 STRING name
30 PRINT "Hello!  What is your name?"
40 LINPUT name
50 Print "Hello!  What is your name?"
60 PRINT "Hello " + name
```

All good programs have one characteristic in common; they are well documented. This means that any person who looks at the program can understand what is happening in various parts of the program. Documentation in CBASIC is done with the commands REM or an exclamation point (!).

Ex. 2

```
    10 REM This is an example of documentation
or
    10 !  This is an example of documentation
```

The REM command is nonexecutable, and therefore does not display anywhere on the screen when the program is running.

The first executable statement to be covered in this manual is PRINT. PRINT displays any type of information on the terminal.

Ex. 3

```
    10 !  This example will print the number 3
    20 LET X=3
    30 PRINT X
```

This example shows how PRINT can be used with a variable.

Ex. 4
```
10 !  This example will print a question
20 PRINT "What's happen'in bro?"
```

Notice that a specific sentence that is needed for a prompt is enclosed with quotation marks.

In CBASIC, all variables and arrays that are used in the program must be defined. These definitions are always found at the beginning of the program. If a variable type is not defined it is assumed to be of type REAL.

INTEGER variable-list

Ex. 5
```
10 !  Practice Example
20 INTEGER x
30 PRINT "How many cars do you own?"
40 INPUT x
```

REAL variable-list

Ex. 6
```
10 !  Another Example
20 REAL x,y
30 PRINT "What was the price of your lunch today?"
40 INPUT x
50 PRINT "What was it yesterday?"
60 INPUT y
```

STRING variable-list

Ex. 7
```
10 !  Last Example
20 STRING mag_name
30 PRINT "What is the name of the magnet?"
40 LINPUT mag_name
```

Variables of any type are limited to 3 letters unless one of the characters between the 4th and 9th position is an underscore. The maximum variable name length is 9 characters. Also, variable names may only begin with alphabetic characters. These rules for variables pertain only to internal variables. External variables, which are devices, do not follow this format.

Legal	Illegal
NAM	NAME
NAME_	3NAM
YOURNAME_	YOURNAMES_

This conveniently brings us to the next step. During a program, when the user is asked to type in a response, an input statement must be used. The type of input statement used depends on the nature of the input. When using numeric input, the INPUT statement is used. When names or "strings" are requested, the LINPUT statement is used. See Examples 5,6, and 7.

It was mentioned earlier that a computer executes one line at a time, and the line numbers allow it to keep track of which line to execute next. Sometimes it is very difficult, and often impossible, to sequentially move straight through a program. The GOTO statement allows the programmer to jump from one part of the program to another. The following program utilizes this statement.

Ex. 8

```
10 !  This program will demonstrate GOTO
20 STRING mag
30 PRINT "What magnet do you want to look at?"
40 LINPUT mag
50 PRINT mag
60 GOTO 30
70 !  End of Program
```

Here the GOTO is used to loop through a number of "would-be" repeated statements.

The next statement that is used quite frequently while programming in CBASIC is the IF-THEN statement. It would generally appear like this:

```
       IF expression 1 THEN expression 2
```

For instance, we might encounter this statement:

```
       IF x=4 THEN GOTO 25
```

This statement is saying, "If the value of x = 4 then the next line to execute is 25. If the value of x is not 4, then the next statement to follow is executed. Also see example 9a.

The next statement that this manual will attempt to explain is FOR...NEXT. This is a very useful command for repeating a series of statements as many times as you like. This is a loop that starts with a FOR statememt and ends with a NEXT statement. The general format of a FOR statement is FOR w=x TO y STEP z. This begins the FOR..NEXT loop, and initializes the variable w to some number x. Then w is incremented each time the NEXT statement is encountered until the numeric value of y is reached. The increment STEP default is 1. Let's compare examples 9a and 9b.

Ex. 9a

```
       10 !  This program will print 1 to 10 using
       20 !  the GOTO command
       30 INTEGER x
       40 x=1
       50 PRINT x
       60 x=x+1
       70 IF (x<11) THEN GOTO 50
       80 !  End of program
```

Ex. 9b

```
       10 !  This program will print 1 to 10
       20 INTEGER x
       30 FOR x=1 TO 10
       40 PRINT x
       50 NEXT x
```

It is important to point out that after the loop is finished, the value of x, in example 9b, is one more than the upper limit (here, x=11). Notice example 9b does exactly what example 9a does, but in fewer steps.

As the computer encounters line 30 (Ex. 9b) for the first time, it is told that until it encounters a NEXT statement the value of x is 1. After it executes line 50, it comes back up to line 30, but now x has a value of 2. It will continue this incrementing process until the last value (10) has gone through the loop.
The printout of this program will look like this:

```
1
2
3
4
5
6
7
8
9
10
```

The last area that will be covered in this manual is statements involving the opening and closing of data files. There are two OPEN statements in CBASIC. An OPEN statement prepares the program to either read from or write to a file. The first that will be discussed is OPEN FOR OUTPUT. Once this statement is executed, any input at the end of a PRINT file# statement will be read into that file. After a file is no longer being used, it must be closed. Therefore the counterpart for any OPEN statement is the CLOSE statement. Example 10 demonstrates the idea of opening a file for input and then closing it.

Ex. 10
```
      10 !  This program will open a file, enter some data,
      20 !  and close the file
      30 INTEGER I
      40 OPEN "Sample.DAT;1" FOR OUTPUT AS FILE#1
      50 FOR I= 1 TO 10
      60 PRINT #1, I
      70 NEXT I
      80 CLOSE #1
```

Note: If a file is not closed properly, no data will be written to it even though it will still be saved. The statement "CLOSE" without any file number specified, closes all open files.

The other type of OPEN is the OPEN FOR INPUT. This will read data that is presently in a file. Instead of using a PRINT file as we did with the OPEN FOR OUTPUT statement, an INPUT file is now used. Let's now use the Sample.DAT;1 (file#1) as our input.

Ex.

```
        10 !  This program will open a file, read data from it
        20 !  and close the file
        30 INTEGER I,J
        40 REAL M
        50 OPEN "Sample.DAT;1" FOR INPUT AS FILE #2
        60 FOR J=1 TO 10
        70 INPUT #2, I
        80 !  A new variable (M= 2 times I) is created
        90 Let M=2*I
        100 PRINT M
        110 NEXT J
        120 CLOSE #2
```

For more commands, statements, and functions please refer to the green booklet titled "Operators Guide" (pg. 36).

Here we list a few mathematical operations that may come up frequently.

CBASIC Term	Meaning
+	Addition
−	Subtraction
*	Multiplication
/	Division
^	Exponent